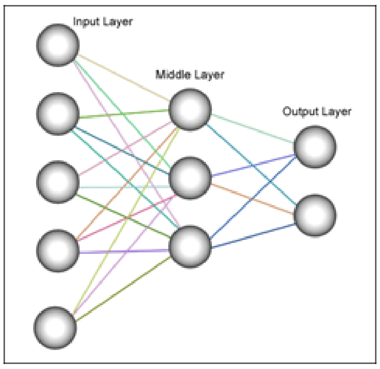
Can the future of application delivery networks be found in neural network theory?

Lori MacVittie, 2008-09-10

I spent a big chunk of time a few nights ago discussing neural networks with my oldest son over IM. It's been a long time since I've had reason to dig into anything really related to AI (artificial intelligence) and at first I was thinking how cool it would be to be back in college just exploring topics like that. Then, because I was trying to balance a conversation with my oldest while juggling my (fussy) youngest on my lap, I thought no, no it wouldn't.



Artificial neural networks (ANN) are good for teaching a system how to recognize patterns, discern complex mathematical relationships, and make predictions based on a variety of inputs. It learns by trying and trying again until the output matches what is expected given a sample (training) data set. That learning process requires feedback; feedback that is often given via backpropagation. Backpropagation can be tricky, but essentially it's the process of determining how far off the output is from the expected output, and then propagating that back into the network so it can essentially learn from its mistakes. Just like us.

If you guessed that this was going to tie back into application delivery, you guessed correctly. An application delivery network is not a neural network, but it often has many of the same properties, such as using something similar to a hidden layer (the application delivery controller) to make decisions about application messages, such as to which server to distribute them and how to best optimize those messages.

More interestingly, perhaps, is the ability to backpropagate errors and information through the application delivery network such that the application delivery network automatically adjusts itself and makes different decisions for subsequent requests. If the application delivery network is enabled with a services-based API, for example, it can be integrated into applications to provide valuable feedback regarding the state of that application and the messages it receives to the application delivery controller, which can then be adjusted to reflect changes in the state of that application. This is how we change the weights of individual servers in the load balancing algorithms in what is somewhat akin to modifying the weights of the connections between neurons in a neural net.

But it's merely a similarity now; it's not a real ANN as it's missing some key attributes and behaviors that would make it one.

When you look at the way in which an application delivery network is deployed and how it acts, you can (or at least I can) see the possibilities of employing a neural network model in building an even smarter, more adaptable delivery network. Right now we have engineers that deploy, configure, and test application delivery networks for specific applications like Oracle, Microsoft, and BEA. It's an iterative process in which they continually tweak the configuration of the solutions that make up an application delivery network based on feedback such as response time, size of messages, and load on individual servers. When they're finished, they've documented an Application Ready Network with a configuration that is configured for optimal performance and scalability for that application that can easily be deployed by customers.

But the feedback loop for this piece is mostly manual right now, and we only have so many engineers available for the hundreds of thousands of applications out there. And that's not counting all the in-house developed applications that could benefit from a similar process. And our environment is not ***your*** environment. In the future, it would awesome if application delivery networks acted more like neural networks, incorporating the feedback themselves based on designated thresholds (response time must be less than X, load on the server must not exceed Y) and tweak itself until it met its goals; all based on the applications and environment unique to the organization.

It's close; an intelligent application delivery controller is able to use thresholds for response time and size of application messages to determine to which server an individual request should be sent. And it can incorporate feedback through the use of service-based APIs integrated with the application. But it's not necessarily modifying its own configuration permanently based on that information; it doesn't have a "learning mode" like so many application firewall and security solutions.

That's an important piece we're missing - the ability to learn the behavior of an application in a specific environment and adjust automatically to that unique configuration. Like learning that in your environment a specific application task runs faster on server X than it does on servers Y and Z, so it always sends that task to server X. We can do the routing via layer 7 switching, but we can't (yet) deduce what that routing should be from application behavior and automatically configure it.

We've come a long way since the early days of load balancing, where the goal was simply to distribute requests across machines equally. We've learned how to intelligently deliver applications, not just distribute them, in the years since the web was born. So it's not completely crazy to think that in the future the concepts used to build neural networks will be used to build application delivery *neural* networks.

At least I don't think it is. But then crazy people don't think they're crazy, do they?